# اصول پردازش تصویر

نیم‌سال اول ۱۳۹۹-۱۴۰۰

مدرس: دکتر مصطفی کمالی تبریزی

دانشکده علوم ریاضی

**تمرین سری چهارم ـ سوال دوم**

نام و نام‌خانوادگی: سیدعلیرضا خادم

شماره دانشجویی: ۹۷۱۰۰۳۹۸

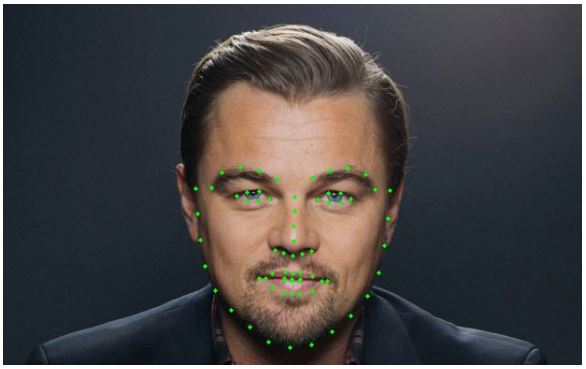
زمان حدودی اجرا: ۱۰ ثانیه برای ۵۰ فریم

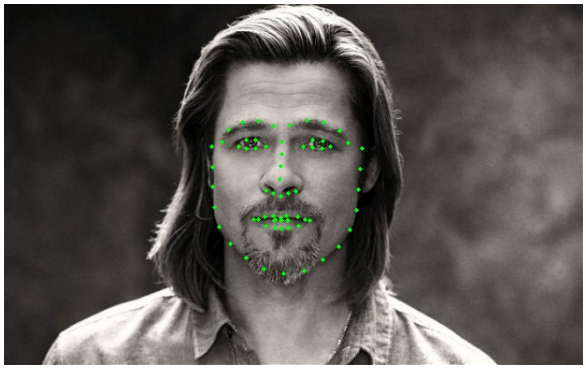
## موارد لازم.

برای اجرا لازم است تا تصاویر BradPitt.jpg و DiCaprio.jpg در مسیر /EX4_Q2 قرار داشته باشد. همچنین در پیاده‌سازی این سوال از کتابخانه‌های cv2 ، dlib ، scipy و numpy استفاده شده است که قبل از اجرا بایستی این کتابخانه‌ها روی سیستم شما نصب باشد.

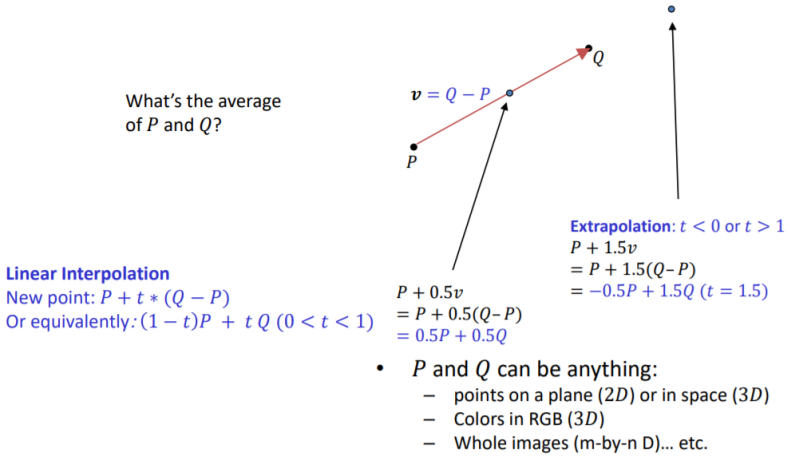
## روند کلی حل.

ایده اصلی حل این سوال به این صورت است که نقاط متناظر بین دو تصویر را بیابیم که در اینجا برای یافتن نقاط متناظرِ صورت دو شخص می‌تواند facial landmarkها را با استفاده از کتابخانه dlib به دست آورد.(این facial landmaarkها برای تصاویر ما به صورت زیر خواهد بود)

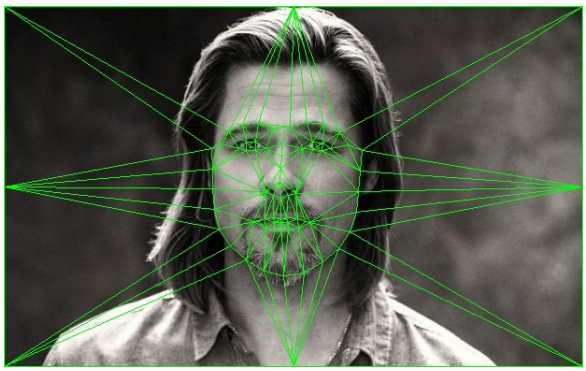


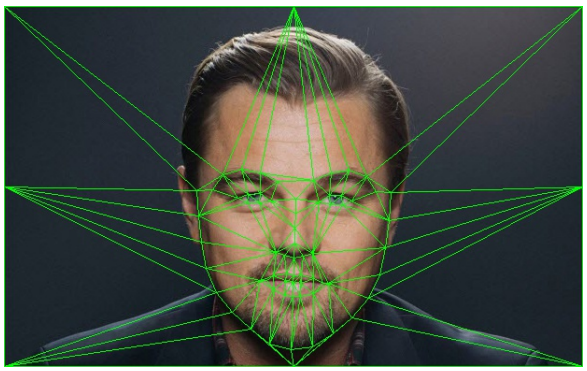


بردار جابه‌جایی بین هر کدام از این key pointها و key point متناظرش در تصویر دوم را محاسبه می‌کنیم و با توجه به تعداد فریم‌های میانی که می‌خواهیم داشته باشیم با استفاده از linear interpolation مختصات key pointها را برای فریم‌های میانی به دست می‌آوریم.




در مرحله بعد روی دو تصویر مثلث‌بندی انجام می‌دهیم. برای مثلث‌بندی از الگوریتم Delaunay که در کتابخانه scipy.spatial پیاده‌سازی شده است استفاده می‌کنیم. فقط باید به این نکته دقت کرد که مثلث‌بندی‌ها در دو تصویر متناظر باشند. برای اینکه مثلث‌بندی‌ها متناظر باشد، یک مثلث بندی روی تصویر اول انجام می‌دهیم و همان مثلث‌بندی را به مثلث دوم منتقل می‌کنیم.همانظور که در تصاویر زیر مشاهده می‌کنید، ۸ نقطه علاوه بر facial landmarks به key points اضافه می‌کنیم تا همه تصویر به صورت کامل مثلث‌بندی شود.





این مثلث‌بندی را برای فریم‌های میانی هم استفاده می‌کنیم. اگر تعداد نقاط متناظر به اندازه کافی زیاد باشد می‌توان بین هر دو مثلث یک نگاشت هندسی ساده در نظر گرفت که این دو مثلث را به هم متناظر تبدیل بکند. با استفاده از رئوس مثلث‌ها نگاشت ها را محاسبه می‌کنیم و همه نقاط داخل مثلث را در فریم‌های میانی را با این نگاشت‌ها از مثلث متناظر در تصویر اول و در تصویر دوم به دست می‌آوریم.

مراحلی که توضیح داده شد به صورت خلاصه در شکل زیر آمده است.

# Summary of Morphing

1. Define corresponding points

2. Define triangulation on points
   - Use same triangulation for both images

3. For each $t = 0 : step : 1$
   a. Compute the average shape (weighted average of points)
   b. For each triangle in the average shape
      - Get the affine projection to the corresponding triangles in each image
      - For each pixel in the triangle, find the corresponding points in each image and set value to weighted average (optionally use interpolation)
   c. Save the image as the next frame of the sequence

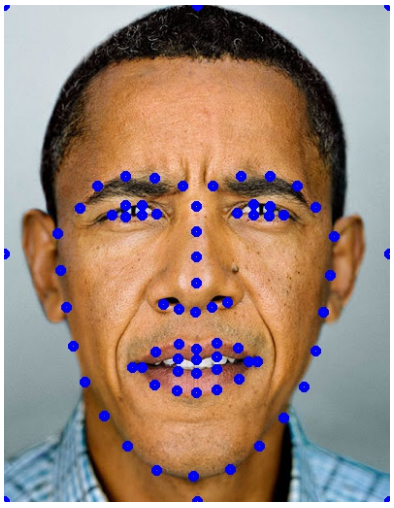
## توضیح کد.

برنامه در مجموع حاوی ۲ فایل با فرمت py. می‌باشد که توضیحات هر فایل در پایین آمده است.

### ○ utilities.py

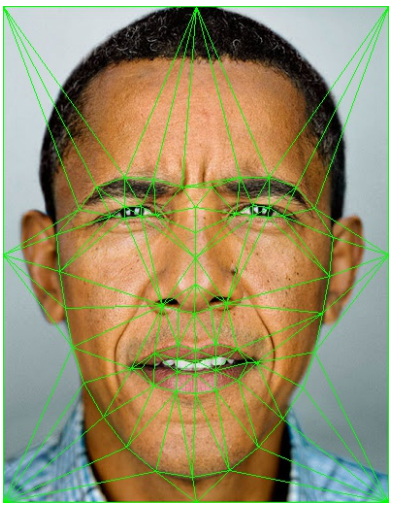
#### get_facial_landmarks(face_image)

این تابع یک تصویر که عکس چهره‌ی یک فرد است را به عنوان ورودی می‌گیرد و با استفاده از کتابخانه‌ی dlib، facial landmarkهایِ چهره به علاوه‌ی ۸ نقطه که مشابه شکل زیر در اطراف تصویر در نظرگرفته می‌شود را به عنوان key points بر می‌گرداند.



**get_triangulation(facial_landmarks)**

این تابع مجموعه نقاطِ facial_landmarks را به عنوان ورودی می‌گیرد و با استفاده از الگوریتم Delaunay که در کتابخانه scipy.spatial پیاده‌سازی شده است یک مثلث‌بندی روی این مجموعه نقاط اعمال می‌کند و این مثلث بندی را به عنوان خروجی تابع برمی‌گرداند. (مشابه تصویر زیر)



**rect_contains(rect, point)**

این تابع یک مستطیل و یک نقطه به عنوان ورودی می‌گیرد و در خروجی مشخص می‌کند که آیا نقطه داخل مستسیل قرار می‌گیرد یا نه.

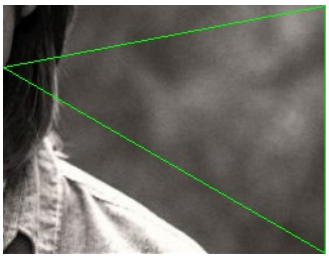
**draw_delaunay_triangulation(src_image, facial_landmarks, triangulation)**

این تابع یک تصویر، یک مجموعه نقاط و یک مثلث بندی روی این مجموعه نقاط را به عنوان ورودی می‌گیرد و این مثلث بندی را روی تصویر رسم می‌کند.

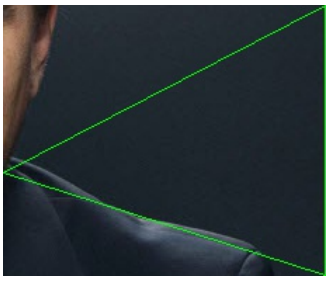
**morph(first_image, last_image, facial_landmarks1, facial_landmarks2, triangulation, Num_of_frames)**

در این تابع یک حلقه به تعداد فریم‌هایی که می‌خواهیم داشته باشیم اجرا می‌شود و در هر بار اجرا یک فریم میانی را تولید می‌کند و در مسیر EX4_Q2/results/ ذخیره می‌کند. به این صورت که روی مثلث‌بندی‌ای که به عنوان ورودی به تابع داشته شده است پیمایش می‌کنیم و در هر پیمایش حلقه‌یِ داخلی یک مثلث را در نظر می‌گیرد و مختصات نقاطِ رئوس این مثلث را در تصویر اول محاسبه می‌کند سپس مستطیلِ محیط بر این مثلث را محاسبه می‌کند و در متغیر face1_triangle_boundary قرار می‌دهد، cropped_face1_triangle_boundary آن مستطیل از تصویرِ اول است که به مثلث در تصویر اول محیط شده است. در ادامه مختصات نقاطِ رئوس این مثلث را در تصویر دوم محاسبه می‌کند سپس مستطیلِ محیط بر این مثلث را در تصویر دوم محاسبه می‌کند و در متغیر face1_triangle_boundary قرار می‌دهد، cropped_face1_triangle_boundary

آن مستطیل از تصویرِ دوم است که به مثلث در تصویر دوم محیط شده است. با استفاده از face1_triangle و face2_triangle و با روش linear interpolation مختصات رئوس این مثلث را در فریم میانی محاسبه می‌کند و در متغیر middle_frame_triangle ذخیره می‌کند.

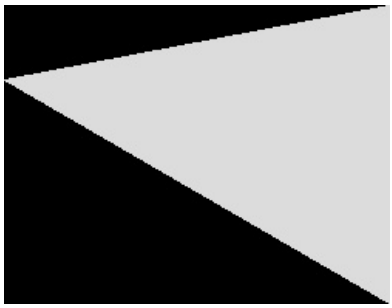


(آ) cropped_face۱_triangle_boundary

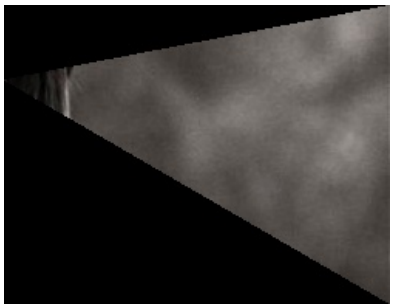


(ب) cropped_face۲_triangle_boundary

با استفاده از تابع fillConvexPloy یک mask برای مثلث میانی همانند تصویر زیر به دست می‌آورد.



در ادامه با استفاده از تابع cv.getAffineTransform نگاشت بینِ مثلث تصویر اول و مثلث میانی و نگاشت بین مثلث تصور آخر و مثلث میانی را به دست می‌آورد و با استفاده از تابع cv.warpAffine این نگاشت‌ها را روی cropped_face1_triangle_boundary و cropped_face2_triangle_boundary اعمال می‌کند و در نهایت cropped_middle_frame_triangle_mask را روی نتایج با استفاده از تابع cv.bitwise_and اعمال می‌کند و نتایج به دست آمده را با روش linear interpolation با هم جمع کرده تا شکلی مشابه تصویر زیر حاصل شود.



در آخر هم آن قسمت از middle_triangle که با تصویری که تا به اینجا ساخته شده اشتراک ندارد را در result قرار می‌دهد.

- **q۲.py**

در این فایل ابتدا تصاویر BradPitt.jpg و DiCaprio.jpg از مسیر /EX4_Q2/images لود می‌شود و در ادامه با استفاده از تابع get_facial_landmarks ، facial landmarkهایِ دو تصویر محاسبه می‌شود و بعد از اینکه مثلث‌بندی محاسبه شد با فراخوانی تابع morph ، morphing انجام می‌شود و نتایج در مسیر /EX4_Q2/results ذخیره می‌شود.